Priebeh celoštátneho kola

Celoštátne kolo 41. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 18.-21. 3. 2026. Na riešenie úloh druhého, praktického dňa majú súťažiaci 4,5 hodiny čistého času. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Skompilovateľný program v podporovanom programovacom jazyku. Ak sa váš program nepodarí na našom testovacom počítači skompilovať a spustiť, bude automaticky hodnotený 0 bodmi.

Odovzdávanie riešení

Riešenia praktického dňa sa odovzdávajú v systéme OIsubmit: https://oi.sk/oisubmit/
Priamo v systéme je pri každej úlohe uvedený presný **časový a pamäťový limit** pre jej riešenia.

Hodnotenie riešení druhého (praktického) dňa

Sú tri úlohy. Ku každej úlohe máme pripravených niekoľko sád testovacích vstupov. Sada vstupov pozostáva z jedného alebo viacerých testovacích vstupov. Za každú sadu vstupov, ktorej každý vstup váš program správne vyrieši, získate v zadaní uvedený počet bodov. Dokopy sa za každú úlohu dá získať 10 bodov.
Testovanie na každom vstupe prebieha samostatne. Spustíme váš program a na štandardný vstup mu dáme konkrétne vstupné údaje. Hovoríme, že váš program daný vstup vyriešil, ak splní nasledujúce kritériá:

- Skončí skôr ako uplynie stanovený časový limit.
- Neprekročí stanovený pamäťový limit.
- Skončí korektne, nie chybou počas behu.
- Dáta, ktoré vypíše na štandardný výstup, tvoria korektný výstup, zodpovedajúci danému vstupu.
- Nebude používať žiadne funkcie zakázané kvôli bezpečnosti testovacieho systému.

Počas súťaže môžete priebežne odovzdávať svoje riešenia. Odovzdané riešenie bude otestované a dozviete sa svoj bodový zisk. Výsledný počet bodov za každú úlohu bude rovný **maximu počtov bodov**, ktoré získali jednotlivé vaše odovzdané riešenia tejto úlohy.
(V prípade preťaženia testovača môžu organizátori obmedziť toto priebežné testovanie na vhodnú podmnožinu všetkých testovacích dát a po skončení súťaže všetky riešenia dotestovať. V prípade technickej chyby, ktorá naruší plánované vyhodnocovanie riešení, môžu organizátori všetky odovzdané riešenia pretestovať.)

A-III-4 Mafiáni u lekára

Tento príbeh sa odohráva v ambulancii v Palerme na Sicílii. Doktorka, ktorá tu ordinuje, práve najala nového recepčného Luku. Luka má za úlohu postupne volať pacientov z čakárne do ordinácie.
Deň ešte len začal, ale Luka už pochopil, že to vôbec nebude jednoduché. Na Sicílii nie je dôležité ani to, kedy kto prišiel, ani to, ako je chorý. Trochu dôležitejšie je, ako je ktorý pacient vplyvný, ale úplne najdôležitejšie je to, z akej je kto rodiny. Nik si nedovolí nechať dlho čakať pacienta z vplyvnej rodiny.

Do čakárne budú počas dňa postupne prichádzať pacienti. Každý pacient bude mať priradené tri nezáporné celé čísla: jeho *poradové číslo* i (za radom 0, 1, 2, atď. v poradí, v ktorom prídu), jeho *vplyv* v_i a jeho *rodinu* r_i.
Luka ešte vôbec nepozná miestne rodiny, takže si o nich priebežne robí názor podľa tých jej členov, ktorých stretne. V každom okamihu si o každej rodine Luka myslí, že jej vplyv je rovný maximu spomedzi vplyvov tých jej členov, ktorých už niekedy videl. (Rátajú sa teda aj tí, čo už boli ošetrení a odišli.) Ak majú podľa Luku dve rodiny presne rovnaký vplyv, za vplyvnejšiu považuje tú s menším číslom.
Raz za čas doktor zavolá ďalšieho pacienta. Vtedy musí Luka niektorého z pacientov v čakárni poslať do ordinácie. Tohto pacienta treba vždy vybrať nasledovne: musí byť z najvplyvnejšej rodiny v čakárni, a ak tam táto rodina má viac členov, tak to musí byť ten, ktorý tam z nich čaká najdlhšie.

Súťažná úloha

Pomôžte Lukovi a napíšte program, ktorý bude podľa vyššie uvedených pravidiel obsluhovať čakáreň.

Formát vstupu a výstupu

Jednotlivé riadky vstupu popisujú jednotlivé udalosti v chronologickom poradí.
Riadok tvaru „0“ popisuje udalosť typu 0: koniec dňa. Takýto riadok je presne jeden, a to posledný.
Riadok tvaru „1 v r“ popisuje udalosť typu 1: príchod nového pacienta s vplyvom v z rodiny r.
Riadok tvaru „2“ popisuje udalosť typu 2: doktor chce ošetriť ďalšieho pacienta.
Pre každú udalosť typu 2 vypíšte jeden riadok a v ňom poradové číslo pacienta, ktorého má Luka poslať do ordinácie, resp. hodnotu -1, ak je momentálne čakáreň prázdna.

Obmedzenia a hodnotenie

Počet udalostí na vstupe si označme q.
V každom vstupe platí, že $q \leq 400\,001$ a pre každého pacienta platí $0 \leq v_i \leq 10^9$ a $0 \leq r_i < 200\,000$.
Existuje niekoľko sád vstupov, každá s nejakými dodatočnými obmedzeniami, ktoré platia pre vstupy v nej. Za každú sadu, ktorú váš program celú korektne vyrieši, dostanete príslušné body.

- Sada 1 (1 bod): Pre každého pacienta platí $r_i = 0$: všetci sú z tej istej rodiny.
- Sada 2 (1 bod): Najskôr všetci pacienti prídu a potom sú nejakí ošetrení – teda udalosti na vstupe majú najskôr typ 1 a potom typ 2.
- Sada 3 (1 bod): Pre každého pacienta platí $v_i = 0$: všetci majú rovnaký vplyv.
- Sada 4 (1 bod): $q \leq 1000$.
- Sada 5 (1 bod): Žiadni dvaja pacienti nie sú z tej istej rodiny ani nemajú rovnaký vplyv. (Všetky v_i aj všetky r_i sú navzájom rôzne.)
- Sada 6 (2 body): Pre každého pacienta platí $r_i < 10$.
- Sada 7 (3 body): Bez dodatočných obmedzení.

Príklad

vstup

```
2
1 1000 3
1 40 2
1 1000 2
2
1 1001 3
2
2
1 47 0
2
1 10 3
2
0
```

výstup

```
-1
1
0
3
2
5
```

- *Lekár zavolal ďalšieho pacienta. V čakárni ale ešte nik nie je, takže nik nejde do ordinácie.*
- *Do čakárne prišiel pacient 0 (vplyv 1000, rodina 3), po ňom pacient 1 (vplyv 40, rodina 2) a po ňom pacient 2 (vplyv 1000, rodina 2).*
- *Lekár zavolal ďalšieho pacienta. Najvplyvnejšia rodina je rodina 2 (obe majú vplyv 1000 a táto má menšie číslo), do ordinácie preto ide jej najdlhšie čakajúci člen: pacient 1.*
- *Prišiel vplyvnejší člen rodiny 3.*
- *Lekár zavolal ďalšieho pacienta. Najvplyvnejšia rodina je momentálne rodina 3, do ordinácie ide preto jej člen: pacient 0.*
- *Lekár zavolal ďalšieho pacienta. Na rad sa dostane ďalší člen rodiny 3.*
- *Prišiel nový pacient z doteraz nevidenej rodiny 0.*
- *Lekár zavolal ďalšieho pacienta. V čakárni máme jedného pacienta z rodiny 2 a jedného z rodiny 0. Vplyvnejšia je rodina 2, na rad sa preto dostane jej ešte čakajúci člen.*
- *Prišiel nový pacient z rodiny 3.*
- *Lekár zavolal ďalšieho pacienta. Pacient, ktorý práve prišiel, je z vplyvnej rodiny 3, a teda ide rovno dnu.*
- *Na konci dňa ostal v čakárni neošetrený pacient s poradovým číslom 4. Ten sa do ordinácie ani na výstup nedostal.*

A-III-5 Voda

Speleológovia pomocou ultrazvuku objavili v Slovenskom Krase novú jaskyňu. Je tvaru zvislého obdĺžnika: široká, hlboká, ale veľmi úzka. Na vstupe dostanete mapu tejto jaskyne: bitmapu tvorenú znakmi „`.`“ (bodka, predstavuje prázdne políčko) a „`X`“ (veľké iks, predstavuje skalu). Jaskyňa je momentálne celá pod zemou a nie je dostupná. Všetky znaky na obvode bitmapy sú preto `X`.
Jaskyňa môže byť sčasti alebo aj úplne zaplavená vodou. Voda sa samozrejme správa ako kvapalina – rozlieva sa do bokov aj dodola a každá súvislá vodou zaplnená oblasť má všade hladinu v tej istej výške. Správanie sa vody bude nižšie zadefinované presnejšie.
Jaskyňa s vodou vznikne tak, že zoberieme jaskyňu a niektoré (možno všetky alebo žiadne) prázdne políčka zmeníme na políčka s vodou.
Voda v takejto jaskyni tvorí nejaké *súvislé oblasti*. Dve políčka s vodou patria do tej istej oblasti ak vie z jedného z nich na druhé preplávať ryba, ktorá sa vie hýbať len vodou a len v smeroch hore, dole, vpravo a vľavo.
Jaskyňa s vodou je *stabilná*, ak platia nasledujúce podmienky:

- Žiadne prázdne políčko nemá hneď nad sebou, vľavo od seba ani vpravo od seba políčko s vodou.
- Ak má prázdne políčko p hneď pod sebou vodu, oblasť, do ktorej patrí táto voda, musí celá ležať v riadkoch nižších ako p.

Súťažná úloha

Pre danú mapu jaskyne spočítajte, koľkými spôsobmi môže byť zaplavená vodou – teda koľko rôznych stabilných jaskýň s vodou z nej vieme vyrobiť.
Presnejšie, nech p je presný počet stabilných jaskýň s vodou, ktoré vieme z danej jaskyne vyrobiť. Toto číslo môže byť veľmi veľké. Aby ste vo svojich programoch nemuseli pracovať s veľkými číslami, je vašou úlohou vypočítať hodnotu $q = (p \bmod 1\,000\,000\,007)$, teda zvyšok, ktorý p dáva po delení $10^9 + 7$.

Formát vstupu a výstupu

V prvom riadku vstupu sú čísla r a s: počet riadkov a stĺpcov bitmapy. Zvyšok vstupu tvorí r riadkov a v každom z nich s znakov. Každý z týchto znakov je „`.`“ alebo „`X`“. Celý obvod bitmapy tvoria „`X`“.

Na výstup vypíšte jeden riadok a v ňom jedno celé číslo: zodpovedajúcu hodnotu q.

Obmedzenia a hodnotenie

Je desať sád vstupov. Za každú sadu, ktorú tvoj program celú korektne vyrieši, je bod. Rôzne sady majú rôzne maximálne veľkosti vstupov a rôzne dodatočné obmedzenia. Údaje o jednotlivých sadách sú v tabuľke nižšie, pričom používame nasledovnú terminológiu:

- *Miestnosť* v jaskyni je maximálna súvislá oblasť tvorená voľnými políčkami.
- Jaskyňa je *súvislá*, ak má len jednu oblasť.
- V *jaskyni so stalagmitmi* platí, že každé políčko, ktoré je `X` a nie je na obvode, má pod sebou ďalšie `X`.
- V *jaskyni so stalaktitmi* platí, že každé políčko, ktoré je `X` a nie je na obvode, má nad sebou ďalšie `X`.
  (Stalagmity sú kvaple rastúce z podlahy a stalaktity sú kvaple visiace zo stropu jaskyne.)

sada	$\max(r, s) \leq$	ďalšie obmedzenia
01	10	jaskyňa je súvislá a jediná miestnosť má tvar obdĺžnika
02	1000	každá miestnosť má tvar obdĺžnika
03	50	jaskyňa je súvislá a ide o jaskyňu so stalagmitmi
04	1000	ide o jaskyňu so stalagmitmi
05	50	jaskyňa je súvislá a ide o jaskyňu so stalaktitmi
06	1000	ide o jaskyňu so stalaktitmi
07	50	jaskyňa je súvislá
08	50	
09	300	jaskyňa je súvislá
10	1000	

Príklady

Prvý príklad vstupu by mohol byť v sade 2 (každá miestnosť je obdĺžnik), druhý v sade 3 (ide o súvislú jaskyňu so stalagmitom), tretí v sade 5 (súvislá jaskyňa so stalaktitmi) a štvrtý v sade 8 (malá jaskyňa, ktorá nepatrí do žiadnej zo skorších sád).

vstup

```
4 9
XXXXXXXXX
X.XX.XXXX
XX.X.X..X
XXXXXXXXX
```

výstup

```
24
```

Táto jaskyňa má štyri samostatné časti. Pre každú jednopolíčkovú časť máme dve možnosti: buď ostane prázdna alebo tam bude voda. Pre zvislú väčšiu časť máme tri možnosti: môže byť buď prázdna, alebo plná do polovice (spodné políčko má vodu, horné nie), alebo je celá plná vody. Vodorovná dvojpolíčková oblasť musí byť buď celá plná alebo celá prázdna.

vstup

```
4 5
XXXXX
X...X
X.X.X
XXXXX
```

výstup

```
5
```

Všetkých 5 možných stabilných jaskýň s vodou je zobrazených nižšie. Znak `O` *predstavuje políčko s vodou.*

```
XXXXX   XXXXX   XXXXX   XXXXX   XXXXX
X...X   X...X   X...X   X...X   XOOOX
X.X.X   XOX.X   X.XOX   XOXOX   XOXOX
XXXXX   XXXXX   XXXXX   XXXXX   XXXXX
```

vstup

```
4 7
XXXXXXX
X.X.X.X
X.....X
XXXXXXX
```

výstup

```
3
```

Všetky riešenia:

```
XXXXXXX   XXXXXXX   XXXXXXX
X.X.X.X   X.X.X.X   XOXOXOX
X.....X   XOOOOOX   XOOOOOX
XXXXXXX   XXXXXXX   XXXXXXX
```

vstup

```
5 9
XXXXXXXXX
X.X.X...X
X...X.X.X
X.X.XXX.X
XXXXXXXXX
```

výstup

```
42
```

A-III-6 Nechaj to na solver III

Toto je open data úloha. Dostanete od nás konkrétne testovacie vstupy. Za každý správny výstup dostanete bod. Výstupné súbory môžete vyrobiť úplne ľubovoľným spôsobom, je na vás, či a ako pri tom využijete ILP solver. Študijný text aj ostatnú dokumentáciu z domáceho kola nájdete v systéme na odovzdávanie riešení.

Podúloha A: investície so synergiami

Text identický s úlohou z domáceho a krajského kola: Máme e eur. Existuje n projektov (očíslovaných od 1 po n), do ktorých ich vieme investovať. Každý projekt buď podporíme alebo nie. O každom projekte vieme, akou sumou s_i ho treba podporiť a akú hodnotu v_i časom dostaneme späť ako výnos, ak ho podporíme.
Medzi projektmi existujú závislosti: napr. projekt na pestovanie bio kapusty sa nevie uskutočniť ak nebude podporený projekt na montáž efektívneho zavlažovania. Takže ak chceme podporiť kapustu, musíme podporiť aj zavlažovanie. Závislostí je z. Pre každú závislosť sú dané dve čísla projektov: ak chceme podporiť projekt $d_{i,1}$, musíme podporiť aj projekt $d_{i,2}$. (Inými slovami, táto závislosť nám zakazuje podporiť $d_{i,1}$ a zároveň nepodporiť $d_{i,2}$.) Závislosti môžu byť úplne ľubovoľné. Špeciálne, ak máme dve závislosti hovoriace, že x závisí na y a že y závisí na x, znamená to, že môžeme buď podporiť oba tieto projekty alebo ani jeden z nich.
Medzi projektmi tiež existujú konflikty: niektoré dvojice projektov si priamo konkurujú a protimonopolný úrad vydal nariadenie, že z každej takej dvojice projektov smieme podporiť najviac jeden. Existuje k konfliktov. Pre každý z nich poznáme čísla $c_{i,1}$ a $c_{i,2}$ projektov, ktoré nesmieme oba podporiť.

Nové v celoštátnom kole: Medzi projektmi tiež existujú synergie a antisynergie. Keď podporíme naraz celú skupinu konkrétnych projektov, občas na tom vieme vďaka pozitívnej interakcii medzi nimi zarobiť viac, a inokedy zase (napr. keď podporíme okrem chovu rýb aj chemickú továreň povyše) podpora konkrétnej skupiny projektov celkovému zisku uškodí.
Formálnejšie, budeme mať q synergii, očíslovaných od 1 po q. Synergia i prináša zisk t_i (ak je toto číslo záporné, ide o antisynergiu) a týka sa ℓ_i projektov. Čísla týchto projektov sú $u_{i,1}, \ldots, u_{i,\ell_i}$. Ak sú všetky tieto projekty podporené, náš zisk sa zmení o t_i, inak sa nič nestane.
Údaje o takýchto projektoch uložíme do textového súboru nasledovne:

- jeden riadok: kapitál a počet projektov — kladné celé čísla e a n
- jeden riadok: sumy potrebné na podporu — kladné celé čísla $s_1, \ldots, s_n$
- jeden riadok: výnosy — kladné celé čísla $v_1, \ldots, v_n$
- jeden riadok: počet závislostí — nezáporné celé číslo z
- z riadkov: popis jednotlivých závislostí — v i-tom z týchto riadkov sú čísla projektov $d_{i,1}$ a $d_{i,2}$
- jeden riadok: počet konfliktov — nezáporné celé číslo k
- k riadkov: popis jednotlivých konfliktov — v i-tom z týchto riadkov sú čísla projektov $c_{i,1}$ a $c_{i,2}$
- jeden riadok: počet synergií — nezáporné celé číslo q
- jeden riadok: výnosy zo synergií — potenciálne aj záporné celé čísla $t_1, \ldots, t_q$
- jeden riadok: počty projektov v synergiách — kladné celé čísla $\ell_1, \ldots, \ell_q$
- q riadkov: popis jednotlivých synergií — v i-tom z týchto riadkov sú čísla projektov $u_{i,1}$ až u_{i,ℓ_i}

Každý vstupný súbor obsahuje najskôr jeden riadok s číslom τ – počtom testovacích vstupov – a následne obsahuje postupne τ testov, každý vo vyššie uvedenom formáte.
Pre každý zo súborov `s1.in` až `s5.in` vyrobte a odovzdajte výstupný súbor `s?.out` obsahujúci τ riadkov – jeden pre každý test na vstupe. V každom riadku výstupu má byť jedno celé číslo: najväčší možný počet peňazí, ktorý v danom teste môžeme mať na konci, ak si správne vyberieme, ktoré projekty podporiť, a ktoré nie.
Jednotlivé vstupné súbory vyzerajú nasledovne:

- 01: $\tau = 100$ testov, v každom $n \leq 400$ projektov, žiadne synergie ($q = 0$, riadky pre t a ℓ sú prázdne).
- 02: $\tau = 100$ testov, v každom $n \leq 400$ projektov, $q \leq 1$ synergia, pre každú synergiu platí $t_1 > 0$.
- 03: $\tau = 100$ testov, v každom $n \leq 400$ projektov, pre každú synergiu platí $\ell_i = 2$.
- 04: $\tau = 100$ testov, v každom $n \leq 150$ projektov, pre každú synergiu platí $t_i > 0$.
- 05: $\tau = 100$ testov, v každom $n \leq 600$ projektov.

V každom teste platí, že sa všetky čísla vrátane správnej odpovede pohodlne zmestia do 32-bitových celočíselných premenných. Konkrétnejšie, súčet e, všetkých v_i a všetkých $|t_j|$ dokopy neprekročí 150 000 000.
V súboroch `s0.in` a `s0.out` je príklad vstupu a jemu zodpovedajúceho správneho výstupu. V prvom teste je optimálne podporiť projekty 1+2 (lepšie ako 1+3 lebo dostaneme bonus zo synergie), práve jeden z dvojice 4+5 (aby sme nedostali zápornú synergiu) a projekty 6+7 (7 chceme, 6 na splnenie závislosti).

Podúloha B: pílime tyče

V obchode so stavebnými potrebami predávajú tyče dĺžky ℓ milimetrov. My potrebujeme tyče s presne rovnakým profilom ale rôznych dĺžok: pre každé i od 1 po n potrebujeme p_i kusov tyčí dlhých d_i milimetrov. Máme dokonalú pílu, na ktorej si vieme obchodné tyče napíliť na ľubovoľné kusy bez akýchkoľvek strát materiálu. Potrebujeme zistiť, koľko najmenej tyčí treba nakúpiť a ako ich popíliť tak, aby sme vyrobili všetko, čo potrebujeme.
Údaje o tomto probléme uložíme do textového súboru nasledovne:

- jeden riadok: dĺžka obchodných tyčí — kladné celé číslo ℓ
- jeden riadok: počet typov tyčí, ktoré potrebujeme — kladné celé číslo n
- jeden riadok: dĺžky tyčí, ktoré potrebujeme — navzájom rôzne kladné celé čísla $d_1, \ldots, d_n$
- jeden riadok: počty tyčí, ktoré potrebujeme — kladné celé čísla $p_1, \ldots, p_n$

Správny výstup k takémuto problému má nasledovný formát:

- jeden riadok: počet obchodných tyčí, ktoré treba nakúpiť — kladné celé číslo m
- jeden riadok: počet inštrukcií na pílenie — kladné celé číslo x
- x riadkov: jednotlivé inštrukcie na pílenie

Každá inštrukcia na pílenie začína kladným číslom: počtom obchodných tyčí, na ktoré ju použiť. Zvyšok riadku potom obsahuje záznamy tvaru „y_i `x` d_i“ hovoriace, že z tyče máme y_i-krát odpíliť kus dĺžky d_i. Namiesto `1x` d_i je tiež povolené vypísať len d_i. Pre každú inštrukciu musí platiť, že súčet dĺžok vyrobených tyčí je $\leq \ell$.
Príklad: aj inštrukcia „`3 1x100 5x40 1x10`“ aj „`3 5x40 10 100`“ nám hovorí, že máme kúpiť 3 obchodné tyče a každú z nich napíliť tak, aby sme dostali jeden kus dĺžky 10, jeden dĺžky 100 a päť dĺžky 40. Takúto inštrukciu samozrejme môžeme použiť len ak majú obchodné tyče dĺžku $\ell \geq 310$.

Výstup je korektný, ak je hodnota m najmenšia možná a následne ostatné riadky obsahujú popis jedného optimálneho spôsobu ako pílením vyrobiť všetky tyče, ktoré potrebujeme, pričom navyše musí platiť $x \leq 1000$ (teda máme nanajvýš 1000 inštrukcií).
Riešenia so správnym m, ktoré okrem potrebných tyčí vyrobia aj nejaké navyše, budú tiež akceptované.

Každý vstupný súbor obsahuje najskôr jeden riadok s číslom τ – počtom testovacích vstupov – a následne obsahuje postupne τ testov, každý vo vyššie uvedenom formáte.
Pre každý zo súborov `t1.in` až `t5.in` vyrobte a odovzdajte výstupný súbor `t?.out` obsahujúci τ blokov riadkov – jeden blok pre každý test na vstupe. Každý blok riadkov musí obsahovať správny výstup pre príslušný test (vo vyššie uvedenom formáte).
Jednotlivé vstupné súbory vyzerajú nasledovne:

- 01: $\tau = 50$ testov, v každom chceme ≤ 38 tyčí a zaručene to ide z $m = 2$ obchodných tyčí.
- 02: $\tau = 50$ testov, v každom chceme ≤ 22 tyčí celkovej dĺžky $> 3\ell$ a ide to z $m = 4$ obchodných tyčí.
- 03: $\tau = 50$ testov, v každom $10^6 \leq \ell \leq 10^7$, vyrábame tyče s dĺžkami $d_i \leq 30$ a celkovou dĺžkou $\leq 15\ell$ a vždy existuje riešenie v ktorom je celková dĺžka odpadu menšia ako 100 000.
- 04: $\tau = 30$ testov, v každom $\ell = 10^7$, $n \leq 36$ a pre každé i platí $d_i > 2\,000\,000$ a $p_i = 1$.
- 05: $\tau = 30$ testov, v každom $\ell = 10^7$, $n \leq 30$ a pre každé i platí $d_i > 2\,000\,000$.

V súboroch `t0.in` a `t0.out` je príklad vstupu a jedného možného zodpovedajúceho správneho výstupu. V prvom teste kúpime dve tyče, z prvej vyrobíme dva kusy dĺžky 334 a z druhej tretí takýto kus. V druhom teste kúpime dve tyče a každú na jednom vhodnom mieste rozpílime. V treťom teste kúpime sedem tyčí. Tri z nich len skrátime na 670 a zvyšok zahodíme. Z dvoch vyrobíme po kuse dĺžky 780 (a z jednej aj zbytočný kus dĺžky 220). Zo šiestej tyče dostaneme kus dĺžky 560 a zo siedmej až dva kusy dĺžky 450 (hoci nám stačí len jeden).

Odovzdávanie riešení

Odovzdajte ZIP súbor obsahujúci ľubovoľnú podmnožinu súborov `s?.out` a `t?.out`, za každý správny súbor dostanete bod.

ŠTYRIDSIATY PRVÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Forišek, Sebastian Hajdu, Paulína Smolárová
Recenzia: Michal Anderle, Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: NIVAM – Národný inštitút vzdelávania a mládeže, Bratislava 2026